Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method of processing instructions in a microprocessor, comprising:

(a) fetching instructions from an instruction memory, certain fetched instructions being load instructions (loads) and causing load operations, and other fetched instructions being store instructions (stores) and causing store operations;

(b) executing the fetched instructions out of program order;

(c) detecting a load/store order violation wherein a load executes prior to a store on whose data the load depends;

(d) creating a store set ~~for the load~~comprising a store set identifier that identifies the store of the load/store order violation, and wherein the store set identifier associates the load with the store;

(e) ~~adding the store to~~ saving the store set to a store set identifier table;

(f) determining whether the store is poisoned by a previously poisoned instruction;

(g) if the store is poisoned, setting a poison value that indicates through a store set dependence that that the store is poisoned; ~~and~~

(h) re-processing said load if said poison value associated with said store indicates through the store set dependence that the store has been poisoned; and

(i) applying said poison value through the store set dependence to subsequent load/store order violation occurrences.

2. (Original) The method of claim 1 wherein (g) includes setting a bit in a table.

3. (Original) The method of claim 1 wherein the store set includes a pointer that points to the poison value.

4. (Original) The method of claim 1 wherein said store set includes a pair of tables which are used to identify said store instruction.

5. (Original) The method of claim 4 further including clearing said poison value when said store is no longer poisoned.

6. (Cancelled).

7. (Cancelled).

8. (Cancelled).

9. (Cancelled).

10. (Currently amended) A computer system, comprising:
a microprocessor comprising a store set identifier table;
an input device coupled to said microprocessor; and
memory coupled to said microprocessor, said memory containing executable instructions;
wherein said microprocessor:
fetches instructions from said memory, certain fetched instructions being load instructions (loads) and causing load operations, and other fetched instructions being store instructions (stores) and causing store operations;
executes the fetched instructions out of program order;
detects a load/store order violation wherein a load executes prior to a store on whose data the load depends;

creates a store set ~~for the load~~comprising a store set identifier value that identifies the store of the load/store order violation, and wherein the store set identifier value links the load to the store;

~~adds the store to~~saves the store set to the store set identifier table;

determines whether the store is poisoned by a previously poisoned instruction;

if the store is poisoned, sets a poison value that indicates through a store set dependence that that the store is poisoned; ~~and~~

re-processes said load if said poison value associated with said store indicates through the store set dependence that the store has been poisoned; and.

applies said poison value through the store set dependence to subsequent load/store order violation occurrences.

11. (Original) The system of claim 10 wherein said poison value comprises a bit in a table.

12. (Original) The system of claim 10 wherein the store set includes a pointer that points to the poison value.

13. (Original) The system of claim 10 wherein said store set includes a pair of tables which are used to identify said store instruction.

14. (Currently amended) The ~~method~~ system of claim 13 wherein said microprocessor clears said poison value when said store is no longer poisoned.

15. (Cancelled).

16. (Cancelled).

17. (Cancelled).

18. (Cancelled).

19. (Currently amended) A microprocessor, comprising:

a fetch stage which fetches executable instructions from memory, certain fetched instructions being load instructions (loads) and causing load operations, and other fetched instructions being store instructions (stores) and causing store operations;

an execution stage coupled to said fetch stage which executes the fetched instructions out of program order;

a store set identifier table coupled to said fetch and execution stages that comprises a store set linked to a load, wherein the store set comprises a store set identifier value that identifies a store of a load/store order violation; and

logic coupled to said fetch ~~and~~ stage, said execution stage, and said store set identifier tables that detects ~~a~~ the load/store order violation wherein ~~a~~ the load executes prior to ~~a~~ the store on whose data the load depends, creates a store set linked to ~~for~~ the load, ~~adds the store to~~ saves the store set to the store set identifier table, determines whether the store is poisoned by a previously poisoned instruction, if the store is poisoned, sets a poison value that indicates through a store set dependence that the store is poisoned, ~~and~~ re-processes said load if said poison value associated with said store indicates through the store set dependence the store has been poisoned, and re-applies said poison value through the store set dependence to subsequent load/store order violation occurrences.

20. (Original) The microprocessor of claim 19 wherein said poison value comprises a bit in a table.

21. (Original) The microprocessor of claim 19 wherein the store set includes a pointer that points to the poison value.

22. (Original) The microprocessor of claim 19 wherein said store set includes a pair of tables which are used to identify said store instruction.

23. (Original) The microprocessor of claim 22 wherein said logic clears said poison value when said store is no longer poisoned.

24. (Currently amended) A microprocessor, comprising:

a fetch stage which fetches instructions including a store instruction (store) and a load instruction (load) that target a common memory location;

a store set identifier table coupled to said fetch stage that comprises a store set associated with said load, wherein said store set comprises a store set identifier that identifies said store of a load/store violation; and

logic coupled to said fetch stage and said store set identifier table, said logic ~~which~~ determines if the data to be written by said store is stale, ~~and~~ if said data is stale, sets a value associated with said store and re-processes said load to execute after said data is no longer stale, establishes said store set for said load to include said store, and saves said value associated with said store in said store set identifier table.

25. (Cancelled).

26. (Currently amended) The microprocessor of claim ~~25~~ 24 wherein said logic uses said store set to access said value.

27. (Original) The microprocessor of claim 24 wherein said value comprises a poison bit.